Indoor Positioning using the IEEE 802.11 Infrastructure

Philippe Jacquet, Yacine Mezali, Georgios Rodolakis

INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Indoor Positioning using the IEEE 802.11 Infrastructure

Philippe Jacquet — Yacine Mezali — Georgios Rodolakis

N° 7294

Mai 2010

Thème COM

ISSN 0249-6399 ISRN INRIA/RR--7294--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
PARIS - ROCQUENCOURT

Indoor Positioning using the IEEE 802.11 Infrastructure

Philippe Jacquet*†, Yacine Mezali ‡† , Georgios Rodolakis

Thème COM — Systèmes communicants
Équipes-Projets Hipercom

Rapport de recherche n° 7294 — Mai 2010 — 33 pages

Abstract: We propose a new indoor localization method which can be used to track a mobile node . Our method uses only the received signal strengths as input information . In addition our approach doesn't require any prior knowledge of the mobile node's motion and therefore doesn't use a cinematic motion model for tracking the mobile node. We discuss in detail the features of our approach and its resulting algorithm .We evaluate the performances of our algorithm using a real signal map..

Key-words: Indoor Positioning, statistical filtering ,IEEE802.11

* Footnote for first author
† Shared foot note
‡ Footnote for second author

Centre de recherche INRIA Paris – Rocquencourt
Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay Cedex
Téléphone : +33 1 39 63 55 11 — Télécopie : +33 1 39 63 53 30

rapport de recherche
Inria

Résumé : Pas de résumé

Mots-clés : localisation indoor, filtrage statistique, infrastructure wifi

1 Localization

1.1 Introduction

Over this last decade, there has been a tremendous growth in the development of WLAN s (wireless local area network). This is mainly due to the development of wireless technologies. The proliferation of these technologies, has contributed to an increasing interests in developing a location aware technologies, ie, devices capable of locating themselves , therefore, providing location aware services. A basic location aware service is the ability of answering the obvious questions where am I? or more complicated questions such as where is the nearest hospital? This problem has several names: location sensing [1], location estimation , location identification , location determination , geo location, it is named radiolocation when it uses a wireless technologies [2].
Systems addressing this problem are known as positioning systems and can be categorized as outdoor positioning systems and indoor positioning systems. This distinction can be considered as a first classification, according to the operating environment of the positioning systems .One of the most well known outdoor positioning system is the GPS(Global Positioning System, the planned Galileo) which are based on signals transmitted from satellites . According to their topologies, the positioning systems can be classified into four categories [1]: Remote positioning systems: the mobile node is the transmitter , whereas several fixed (anchored) nodes receive ,and measure the transmitter's signal .
The measures are collected by anchored nodes ,and the transmitter's location is computed in a master station [1]. Self positioning systems: the mobile node computes its own location by receiving the signals of different (anchored) transmitters whose locations are known Indirect remote positioning systems: these systems differ from the self positioning system class by the presence of a wireless data link which is used to send the measurements result from the self positioning node to the remote side. Indirect self positioning systems: these systems differ from the remote positioning systems, by their ability to send the measurements result to the mobile node via wireless data link. Unlike outdoor geolocation , indoor geolocation can be more challenging . In fact, indoor environments contain many obstacles such as : human bodies , floor, ceiling..etc, which , alter the propagation of the underlying medium (electromagnetic waves ,ultrasonic waves ..)signals because of (reflexion , refraction ,and scattering of signals). Indoor gelocation systems can be classified according to : 1-The location positioning algorithm .
2-The underlying wireless technologies which are used to communicate(WIFI, blue tooth, ultrasonic technology....etc [1]. This location positioning algorithms can fall into two categories : the first category involves algorithms based on the traditional concepts such trilateration or triangulation, and the second category is more concerned with algorithms including the signals map concept. The algorithms based on the traditional concepts , exploit specific information such us : time of flight TOF t, angle of arrival AOA ,direction of arrival DOA . These techniques are reliable only when the line of sight (LOS) is dominant [3] which is not the case in an indoor environment. Moreover these techniques require a specific hardware ,and accurate synchronization between transmitters ,and receivers [3].Furthermore , improving the accuracy of a positioning systems , based on the existing network infrastructure , would be less expensive [2]. The

signals map concept aims to build a signal strength model. This model can be either deterministic or probabilistic. The location method tries to extract from this model the best match which can be used to estimate the mobile node position. The class of indoor positioning algorithms, which rely on the signals map concept includes algorithms using deterministic techniques ,and those which are based on the probabilistic approach [2].
In algorithms using deterministic approach , the signals map concept, is implemented with a radio map which is coupled with a matching method. The radio map constitutes a data base (DB) with entries corresponding to average RSS at specific position.
Each entry, in th DB consists of a tuple (RSS_j, P_i) where RSS_i is the average value of the received signal strength from node j at the position P_i.
The matching methods, compute the best match, using the minimal distance, between RSS vector which is measured by the mobile node, ,and all the RSSs vectors in the DB. The distance, can be either the euclidean distance or the Mahalanobis distance .The NNSS (Nearest neighbor in signal space method), for example RADAR Microsoft [4], chooses the best match according to the minimal euclidean distance; the coordinates of the mobile node will be values of the best matching coordinates. Instead of choosing one neighbor , the NNSS AVG (Nearest neighbor in signal average) [8] chooses K neighbors (candidates) ,and computes the coordinates of the mobile node by averaging the coordinates of the K candidates. Although efficient, the (NNSS ,and NNSS AVG) techniques remain expensive in terms of storage capacity as they require a large DB to cover the whole indoor area. The probabilistic approach considers the signals received , from several anchored nodes , as a multivariate density distribution ,and instead of computing the best match (location) deterministically, it computes the probability to be at a certain location given the RSS vector measured by the mobile node [2] .In this case, the data base contains a set of signals samples, collected at certain positions. These samples are used to reconstruct the density distribution of the RSS vector given a certain position[2]. We are more interested in indoor self positioning systems based on IEEE802.11, which rely on a signals map. Our approach falls under probabilistic techniques ; our method, works under the assumption which consists of considering the received signals as a Gaussian vector whose parameters (mean vector , covariance matrix) are function of the position in the radio map.

1.2 Problem Statement

We assume k fixed nodes which are called anchored nodes. These nodes are similar to the *Access Points* of the infrastructure mode which is used in wifi networks [5]. We consider a mobile node A. We denote by $v(t)$ the signal level sampling vector at the time t : $v(t) = (v_1(t), v_2(t), v_3(t), .., v_k(t))$ where each component $v_i(t)$ represents the signal strength level in (dB) received by the mobile node A from the anchored node AP_i at time t, figure 1. The basic *localization problem* consists of determining the mobile node position $z((x(t), y(t))$ at time t given a signal level vector $v(t)$. Our objective is to determine the node's path $C = (z(0), z(1), ., z(t), ..z(T))$, $z(t)$ is node's position in the plane P at time t.

1.3 The Model

We will assume that at each point $z \in P$ the signal level vector is a Gaussian vector $v \sim \mathfrak{N}(m(z), Q_z^{-1})$ where : $m(z) = \mathbf{E}v$ is the expectation vector and $Q_z^{-1} = \mathbf{E}(v - m(z))(v - m(z))^T$ is the covariance matrix, therefore Q_z corresponds to the inverse covariance matrix; the Gaussian vector v is distributed according to the density function :

$$p(z,u) = \sqrt{\frac{\det(Q_z)}{\pi^k}} \exp\Big\{-\frac{1}{2}(u - m(z))^T Q_z (u - m(z))\Big\}$$

which can be also expressed as :

$$p(z,u) = \sqrt{\frac{\det(Q_z)}{\pi^k}} \exp\Big\{-\frac{1}{2}\langle (u - m(z)), Q_z (u - m(z))\rangle\Big\}$$

where $\langle a, b\rangle$ denotes the dot product of two vectors a and b. As our objective is to determine the node's path $C = (z(0), z(1), ., z(t), ..z(T))$ we need to understand what characterizes the most likely path taken by the node A : When a mobile node A is at a certain position z, its distribution is given by the probability density function $p(z,u)$. It is obvious that the density of probability of $v(t)$ computed considering another position $\tilde{z}$ using $p(\tilde{z}, v)$ replacing z by $\tilde{z}$, is less important compared to one computed using $p(z,u)$. So we can deduce that the path $C = (z(0), z(1), ., z(t), ..z(T))$ corresponds to the one that maximizes the product

$$\prod_{i=0}^{i=T} p(z(i), v(i)) \tag{1}$$

if we denote $l(z,v) = -\log(p(z,v))$ maximizing (1) is equivalent to minimizing the sum $\sum_{i=0}^{i=T} l(z(i), v(i))$ or the integral

$$\int_0^T l(z(t), v(t))dt \tag{2}$$

1.4 Path tracking (Localization)Algorithm derivation

In this section we describe the methodology we have used to derive our *localization algorithm*.We introduce an algorithm that aims at each step to correct the discrepancy between the real and the estimated positions.We assume that the mobile node's speed is bounded and we also assume that the perturbation affecting the mobile node's position at a time t is small with respect to the most likely path.Even when a small perturbation in the node's position occurs, we always need to consider the most likely path so we have to minimize (2).
When a small perturbation $\kappa(t)$ affects the mobile node's position $z(t)$ (2) at time t we obtain :

$$\begin{aligned}\int_0^T l(z(t) + \kappa(t), v(t))dt &= \int_0^T l(z(t), v(t))dt \\ &+ \int_0^T \langle \nabla_z l(z(t), v(t)), \kappa(t)\rangle dt\end{aligned} \tag{3}$$

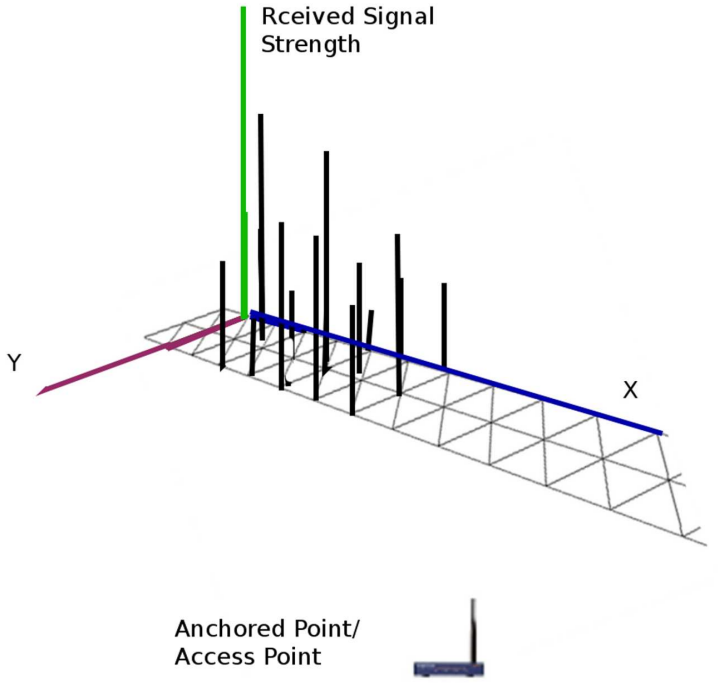


Figure 1: At each position $z = (x, y)$ the RSS is Gaussian whose parameters depend on z

Where $\langle \nabla_z l(z(t), v(t)), \kappa(t) \rangle$ is the dot product of $\nabla_z l(z(t), v(t))$ and $\kappa(t)$. Therefore if we want to optimize the integral (3) without constraints we should have $\langle \nabla_z l(z(t), v(t)), \delta(t) \rangle = 0$ point-wise for all possible $\kappa(t)$ this means that :

- $\nabla_z l(z(t), v(t)) = 0$ or
- $m(z(t) = v(t))$.

The last condition would imply that $z(t)$ will follow locations where $m(z(t)) = v(t)$; which may imply an unbounded speed involving unrealistic motion which makes our assumption for the bounded speed more plausible (realistic). We maintain this assumption in the follwing sections.

1.5 Path optimization with bounded speed

As we have already assumed the module of the velocity vector is be bounded, therefore we can assume that at any given point z, the module of the speed vector $\|\vec{\dot{z}}\|$ is constant with only direction as tunable parameter; in this case we can also have the additional condition that the speed vector $\vec{\dot{z}}$ is orthogonal to the perturbation vector $\delta(t)$. In order to satisfy the condition $\langle \nabla_z l(z(t), v(t)), \delta(t) \rangle = 0$ point-wise, we should have $\nabla_z l(z(t), v(t))$ proportional to $\vec{\dot{z}}$. As stated by the assumptions and conditions above :

$$z(t + \zeta(t)) = z(t) + \zeta(t)\dot{z} \tag{4}$$

$$z(i + 1) = z(i) + \dot{z} \tag{5}$$

Consequently a path such that : $z(i + 1) = z(i) + C_z \nabla_z l(z(i), v(i))$ or with an arbitrary time step $\zeta(t)$: $z(t + \zeta(t)) = z(t) + \zeta(t) C_z \nabla_z l(z(t), v(t))$ where $\nabla_z l(z(t), v(t))$ is proportional to $\vec{\dot{z}}$ is a good candidate for our path tracking algorithm; as $v(t)$ is random, the evolution of $z(t)$ could be regarded as a random walk. In order to investigate the evolution of this random walk, we compute the gradient expression :

$$\begin{aligned} \nabla_z l(z, v) = \frac{1}{2} (& \langle (v - m(z)), \langle \nabla Q_z, (v - m(z)) \rangle \rangle \\ & - 2 \langle \nabla_z m(z), Q_z (v - m(z)) \rangle \\ & - \nabla_z \log \det(Q_z)) \end{aligned} \tag{6}$$

Notice that $\log \det(Q_z) = tr(\log(Q_z))$ and therefore $\nabla_z \log \det(Q_z) = tr(Q_z^{-1} \nabla_z Q_z)$. We also have $\langle u Q_z u \rangle = tr(Q_z u \otimes u)$ or $u^T Q_z u = tr(Q_z u \otimes u)$ where $u \otimes u$ is the Kronecker product.

1.5.1 Random Walk optimization

Since the candidate path evolves in a random walk, it would be interesting to compute the average drift of this random walk, in order to calculate the constant C_z Assume that v is normally distributed with an inverse covariance matrix Q and mean vector m i.e :

$$E(v) = m$$

$$E((v - m) \otimes (v - m)^T) = Q^{-1}$$

Then we have :

$$\begin{aligned} E(\nabla_z l(z,v)) &= \frac{1}{2} E(\langle (v-m(z)), \langle \nabla Q_z, (v-m(z)) \rangle \rangle \\ &- 2\langle \nabla_z m(z), Q_z(v-m(z)) \rangle \\ &- \nabla_z \log\det(Q_z)) \end{aligned} \tag{7}$$

which leads to :

$$\begin{aligned} E(\nabla_z l(z,v)) &= \frac{1}{2} (E(\langle (v-m(z)), \langle \nabla Q_z, (v-m(z)) \rangle \rangle) \\ &- 2E(\langle \nabla_z m(z), Q_z(v-m(z)) \rangle) \\ &- \nabla_z \log\det(Q_z)) \end{aligned} \tag{8}$$

which yields :

$$\begin{aligned} E(\nabla_z l(z,v)) &= \frac{1}{2} (\langle E(v-m(z)), \langle \nabla Q_z, E(v-m(z)) \rangle \rangle \\ &- 2\langle \nabla_z m(z), Q_z E(v-m(z)) \rangle \\ &- \nabla_z \log\det(Q_z)) \end{aligned} \tag{9}$$

Since $\nabla_z \log\det(Q_z) = tr(Q_z^{-1}\nabla_z Q_z)$ we obtain:

$$\begin{aligned} E(\nabla_z l(z,v)) &= \frac{1}{2} (tr(Q_z Q^{-1} \nabla_z) + \langle (m-m(z)), \langle \nabla Q_z, (m-m(z)) \rangle \rangle \\ &- 2\langle \nabla_z m(z), Q_z(m-m(z)) \rangle \\ &- tr(Q_z Q^{-1} \nabla_z Q_z)) \end{aligned} \tag{10}$$

We notice that the term $\frac{1}{2} tr(\nabla_z Q_z Q^{-1})$ has been added to (10) to cancel $-tr(\nabla_z Q_z Q_z^{-1})$ when $Q = Q_z$ and $m = m(z)$ which is the case when v is exactly distributed like the sampled signal at position z Now we assume that a small perturbation $\dot{z}\zeta(t)$ in the node's position z affects its statistic parameters $m(z)$ and Q_z (the mean vector and the inverse covariance matrix)as follows :

$$\begin{aligned} m &= m(z + \zeta(t)\dot{z}) = m(z) + \langle \dot{z}\nabla_z m(z) \rangle \zeta(t) \\ Q &= Q(z + \zeta(t)\dot{z}) = Q_z + \langle \dot{z}\nabla_z Q_z \rangle \zeta(t) \end{aligned} \tag{11}$$

From (11) we have :

$$m - m(z) = \langle \dot{z}\nabla_z m(z) \rangle \zeta(t) \tag{12}$$

$$Q - Q_z = \langle \dot{z}\nabla_z Q_z \rangle \zeta(t)$$

Where Q and Q_z are the inverse covariance matrices at positions $z + \zeta\dot{z}$ and z respectively. After substituting (12) in (10) we obtain :

$$\begin{aligned} E(\nabla_z l(z,v)) &= \frac{1}{2} (tr((Q^{-1} - Q_z^{-1})\nabla_z Q_z)\zeta(t) \\ &+ \langle \nabla_z \langle m(z)\dot{z}\zeta(t) \rangle, \langle \nabla Q_z, \langle \nabla_z m(z)\dot{z}\zeta(t) \rangle \rangle \rangle \\ &- 2\langle \nabla_z m(z), Q_z \langle \nabla_z m(z)\dot{z}\zeta(t) \rangle \rangle) \end{aligned} \tag{13}$$

Let us now develop $Q^{-1} - Q_z^{-1}$ at the first order : let us try to develop Q^{-1} or more precisely $(Q_z + \langle \nabla_z Q_z \rangle \zeta(t))^{-1}$ we have :

$$\begin{aligned}(Q_z + \langle \nabla_z Q_z, \dot{z}\zeta(t) \rangle)(Q_z^{-1} - Q_z^{-1}\langle \nabla_z Q_z \dot{z}\zeta(t) \rangle Q_z^{-1}) = \\ I + \langle \nabla_z Q_z, \dot{z}\zeta(t) \rangle Q_z^{-1} - \langle \nabla_z Q_z, \dot{z}\zeta(t) \rangle Q_z^{-1} - \\ \langle \nabla_z Q_z, \dot{z}\zeta(t) \rangle Q_z^{-1} \langle \nabla_z Q_z, \dot{z}\zeta(t) \rangle Q_z^{-1}\end{aligned}$$

where I is the identity matrix. Since we are interested in developing $Q^{-1} - Q_z^{-1}$ at the first order, the last term can be ignored.
Therefore :

$$(Q_z + \langle \nabla_z Q_z, \dot{z}\zeta(t) \rangle)(Q_z^{-1} - Q_z^{-1}\langle \nabla_z Q_z \dot{z}\zeta(t) \rangle Q_z^{-1}) \approx I$$

Then we have :
$Q^{-1} = (Q_z^{-1} - Q_z^{-1}\langle \nabla_z Q_z \dot{z}\zeta(t) \rangle Q_z^{-1})$
Finally :

$$Q^{-1} - Q_z^{-1} = -Q_z^{-1}\langle \nabla_z Q_z \dot{z}\zeta(t) \rangle Q_z^{-1} \tag{14}$$

After substituting (14) in (13) and after limiting the development of terms at the first order we finally derive the average drift :

$$\begin{aligned}E[\nabla_z l(z,v)] = -\frac{\zeta(t)}{2}(tr(\nabla_z Q_z Q_z^{-1} \nabla_z Q_z Q_z^{-1}) + 2(\nabla_z m(z) Q_z \langle \dot{z}, \nabla_z m(z) \rangle)) \\ + O(\zeta(t)^2)\end{aligned} \tag{15}$$

If we assume that the covariance matrix Q_z^{-1} is diagonal, which implies that Q_z is also diagonal.This means that :

$$Q_z^{-1} = \begin{bmatrix} \delta(z)_1 & . & . & \cdots & . \\ . & \delta(z)_2 & . & \cdots & . \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \vdots & \cdots & \delta(z)_K \end{bmatrix}$$

$$Q_z = \begin{bmatrix} \lambda(z)_1 = \frac{1}{\delta(z)_1} & . & . & \cdots & . \\ . & \lambda(z)_2 = \frac{1}{\delta(z)_2} & . & \cdots & . \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \vdots & \cdots & \lambda(z)_K = \frac{1}{\delta(z)_K} \end{bmatrix}$$

Then we can rewrite $E[\nabla_z l(z,v)]$ as follows :

$$E[\nabla_z l(z,v)] = -\frac{1}{2}(G + 2(Q_z \nabla_z m(z) \otimes \nabla_z m(z)^T))\dot{z}\zeta(t) + O(\zeta(t)^2) \;\; With$$

$$G = \sum_{i=1}^{K}\Big(\frac{1}{\lambda(z)_i^2}\nabla_z \lambda(z)_i \otimes \lambda(z)_i^T\Big)$$

Finally :

$$E[\nabla_z l(z,v)] = R\dot{z}\zeta(t) + O(\zeta(t)^2)$$

And

$$R_z = -\frac{1}{2}(G + 2(Q_z \nabla_z m(z) \otimes \nabla_z m(z)^T)) \tag{16}$$

1.6 Localization Algorithm

As we have already seen in section 1.5, a good candidate for the localization algorithm is an algorithm evolving in a random walk :
$z(t+\zeta(t)) = z(t) + \zeta(t) C_z \nabla_z l(z(t), v(t))$
we can notice that $E(R_z^{-1} l(z,v)) = \dot{z}\zeta(t)$ hence, to achieve the correct drift; a good strategy for a localization algorithm of that kind, consists of taking at each step C_z equal to the inverse of the matrix R_z. Therefore, the localization algorithm we propose is the following :

$$z(t+\zeta(t)) = z(t) + R_z^{-1} l(z(t), v(t)) \tag{17}$$

Which computes the node's position $z(t+\zeta(t))$ based on the previous position $z(t)$, Consequently in order to perform localization of the mobile node, we need to compute at each step : R_z and $l(z,v)$ according to (16) and (6).

Algorithm 1 Localization

$n \leftarrow 0$

while true **do**
  Receive $\boldsymbol{V[1:K]}$
  $\boldsymbol{Interpolate_{vector}}\ m(\tilde{z_n})$, $\boldsymbol{EstimateDifferential_{vector}} \nabla_z m(\tilde{z_n})$
  $\boldsymbol{Interpolate_{matrix}}\ Q(\tilde{z_n})$, $\boldsymbol{EstimateDifferential_{matrix}} \nabla_z Q(\tilde{z_n})$
  $\boldsymbol{\nabla_z l(z,v)} \leftarrow \frac{1}{2}(\langle (v - m(z)), \langle \nabla Q_z, (v - m(z)) \rangle \rangle - 2\langle \nabla_z m(z), Q_z(v - m(z)) \rangle - \nabla_z \log\det(Q_z))$
  $\boldsymbol{R_z} = \longleftarrow -\frac{1}{2}(G + 2(Q_z \nabla_z m(z) \otimes \nabla_z m(z)^T))$
  $\boldsymbol{\tilde{z_{n+1}} = \tilde{z_n} + R_z^{-1} \nabla_z L(z,v) \zeta(t)}$
  $\boldsymbol{\tilde{z_n} = \tilde{z_{n+1}},\ n \leftarrow n+1}$
end while

Where $\zeta(t)$ represents the time step.

1.7 Computation of $m(z)$, $\nabla_z m(z)$, $Q(z)$, $\nabla_z Q(z)$ at each step

Since we are trying to locate the mobile node in the plane $\Re^2$, it is almost impossible to store all the $m(z)$ and Q_z at every position z; so we can only afford to measure these parameters $m(z)$, $Q(z)$ for a finite set.We recall that $m(z)$ and Q_z are both functions of the position z but whose analytical expressions are unknown or hard to find. As we have already stated, we can only store a finite number of couples $S = (position_i,\ parameters_i)$ or $(z_i, m(z_i), Q(z_i))$.Our next problem is to introduce a mathematical material which permits the reconstruction of such spatial functions as : $m(z)$ and Q_z. The term spatial stresses the dependency with a respect to the position z, the term reconstruction implies estimating the function values at every point of the plane and computing the gradient at every position z which means studying and implementating the following primitives: $\boldsymbol{Interpolate_{vector}}()$, $\boldsymbol{Interpolate_{matrix}}()$ and $\boldsymbol{EstimateDifferential_{matrix}}()$
$\boldsymbol{EstimateDifferential_{vector}}()$ is the aim of the next chapter.

2 Introduction

Interpolating scattered data is a very applicable technique in a number of scientific domains : geophysics (seismic tomography), astronomy (magnetic fields) and in the field of engineering in such fields as geometric modeling and the numerical resolution of differential equations.

2.1 Problem formulation

The interpolation problem can formulated as follows [6] : Let $S = \{p_1, p_2, .., p_i, ..p_n\}$ denote a set of fixed points in the d-dimensional Euclidian space $\Re^d$ each element of S is called a data site.We note $CH(S)$ the convex hull of S and suppose that at each point p_i a real value $z(p_i)$ is given . The aim of the interpolation problem is to find a function Ψ such that $\Psi(p_i) = z_i$, for $\{i = 1, ..n\}$ and $\Psi(p) = z(p)$ for any point p inside CH(S).The value $z(p)$ is called the interpolant at the target point p. Many approaches may be adopted to solve the interpolation problem, some of these approaches are based on the barycentric coordinates of the target point, the finite element method being one of them [7]. Another approach is based on triangulation spanning, that is to triangulate the $CH(S)$ of S, among the interpolation methods inspired by this approach: interpolating data using Delaunay triangulation, also known as *Voronoi based methods.*

2.2 (Background)The Voronoi Diagram and Delaunay Triangulation

Considering $S = \{p_1, p_2, .., p_i, ..p_n tobe \in \Re^d\}$to be the same set of fixed points defined previously, $p_1, p_2, .., p_i, ..p_n, .$ the elements of S are called the generators, we denote by $d(.., ..)$ the Euclidian distance in $\Re^d$. *The Voronoi diagram or the Voronoi tessellation* of S noted $Vor(S)$ is the partition of $\Re^d$ into convex tiles by the following rule "which generator is the nearest" [6] Formally *the Voronoi diagram* is a collection of Vornoi regions or Voronoi cells as shown in Figure(2). *A Voronoi region* $V(P_i)$ is defined as follows : $V(P_i) = \{P \in \Re^d | d(P, P_i) < d(P, P_j), for \ \ j \neq i\}$ The Voronoi region$V(P_i)$ can be also seen as an intersection of *halfspaces* $H(P_i, P_j)$ bounded by the hyperplanes $\|P - P_i\| = \|P - P_j\|$ where $H(P_i, P_j) = \{P \in \Re^d | d(P, P_i) < d(P, P_j)$. Therefore $V(P_i) = \{\bigcap_{j \neq i} H(P_i, P_j), j \neq i\}$ [8].

The Euclidian space is then partitioned into a complex cell $V(P_1), V(P_2)$, $.., V(P_n..)$. In the d-dimensional Euclidian space $\Re^d$, The boundary of two adjacent regions $V(P_i), V(P_j)$ is a part of the $(d-1)$ dimensional space. For instance : in the two-dimensional space $\Re^2$, a boundary can be either a point or a line segment The dual cell complex of the *Voronoi diagram* is called *Delaunay triangulation* if two cells $V(P_i)$ and $V(P_j)$ in the Voronoi diagram share a common boundary then the generators P_i and P_j are said to be neighbors, *Delaunay triangulation* is the relation which connects each generator to its neighbors and partions the space into convex polygons (Figure3) , triangles in the case of the two-dimensional space $\Re^2$. Delaunay triangulation and Voronoi diagram are said to be dual one to the other, which means that once one is known the other is completely defined. Triangles of Delaunay triangulation satisfy the following properties;

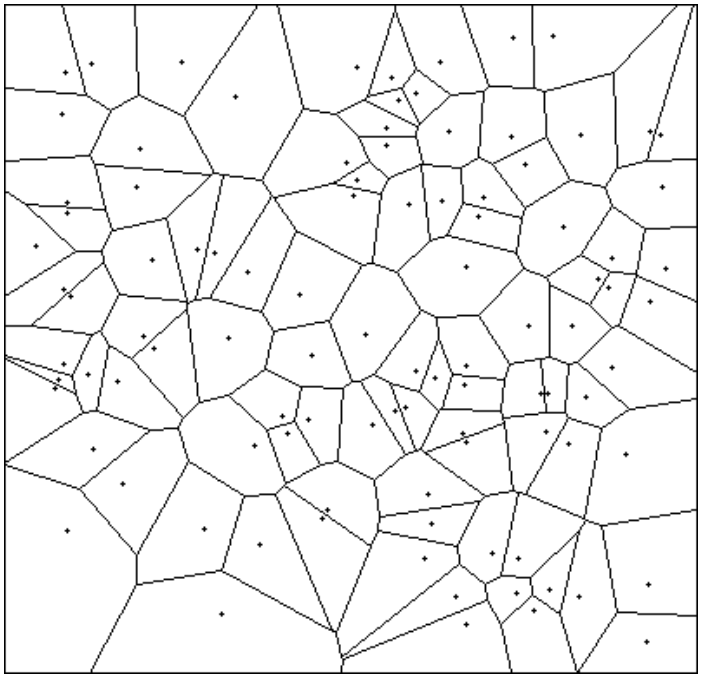


Figure 2: A Voronoi diagram

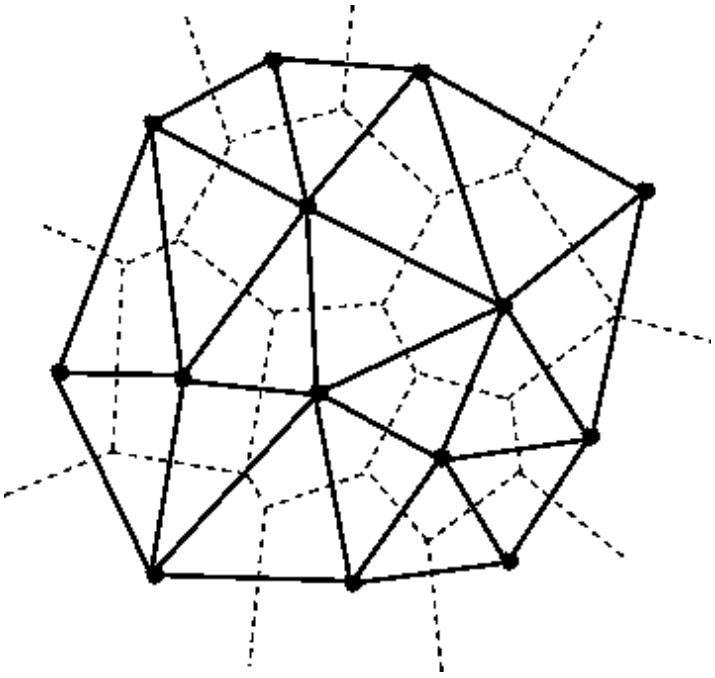


Figure 3: Delaunay Triangulation

- `The Circumcircle property` : No generators inside the circumcircle of any triangle.
- `Equi-angularity` : among all the Triangulation spanning S Delaunay triangulation is the one which maximizes the smallest angle of all triangles in the triangulation.

3 Voronoi Based Interpolation methods [6]

In the following subsections, we shall assume that our generators are located in the two-dimensional space $\Re^2$. S denotes the set of generators $S = \{p_1, p_2, .., p_i, ..p_n\}, p_i \in \Re^2$,$DT(S)$ denotes the Delaunay triangulation of S. We note $z_1, , , z_n$ the data values associated with P_1, P_n and we note $Vor(S)$ as the Voronoi diagram of S. The $DT(S)$ implies that any point $P(x, y)$ inside the convex hull of S can be located in a specific triangle of the $DT(S)$. Therefore $\forall p \in CH(S), \exists \triangle T_l \in DT(S) \mid p \in T_l$. We note $P_1(x_1, y_1), P_2(x_2, y_2), P_3(x_3, y_3)$ as the the vertices of T_l‘.

3.1 Natural Neighbor Coordinates, Sibson’s Interpolant [8]

Assuming we want to evaluate the value of the target point x, we require that P is an inner point to $CV(S)$. Let $S' = S \cup \{x\}$, to evaluate $z(x)$ Sibson uses the Voronoi diagram of S' because of the proximity information among the generators provided the triangulation. [6]. The data sites whose Voronoi regions are adjacent to the Voronoi region of the target point P are interpreted as the data sites nearest the target point (Figure4). The Voronoi diagram of S' is generated from $Vor(S)$ by inserting x into $Vor(S)$. Let $\pi(x)$ denote the potential area of the Voronoi cell of x and let $\pi_i(x)$ denote the overlapping area that will be stolen from the cell of P_i by the cell of x (Figure4).
The natural neighbor coordinate of x with a respect to data site P_i is defined as :

$$\lambda(z)_i(x) = \frac{\pi_i(x)}{\pi(x)}$$

The natural neighbor coordinates satisfy the following properties [7] [9];

- $x = \sum_i \lambda(z)_i(x) P_i$ (The Barycentric property).
- $\forall i, j \leqq n \;\; \lambda(z)_i(P_j) = \delta_{i_j}$, where δ_{i_j} is the Kronecker symbol;

$$\delta_{i_j} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

.

- $\sum_{i=1}^{n} \lambda(z)_i(x) = 1$ (The partition of unity property).

Sibson's interpolant in its most general form is given by;

$$z(x) = \sum_i \lambda(z)_i(x) z_i$$

Although the definition of the neighborhood is local which is an advantage for the method, Sibson's interpolation has some drawbacks : *Main drawbacks of Sibson's interpolation* :

- Sibson's interpolant is C^0 at data sites.
- The natural neighbor coordinates are defined only inside the convex hull of the data sites.

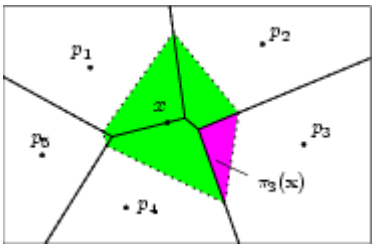


Figure 4: Insertion of the target point x in the Voronoi diagram of S

3.2 Simple Linear Interpolation

The overall aim of this method is to derive a formula that can interpolate an affine function value $z(P)$ for a given point $P(x, y)$ and to compute the gradient vector at this point. In our case, the interpolated value is the *Received Signal Strength RSS* of an *Access Point* AP. The interpolation function $\Psi(p)$ is assumed to be an affine function over $\Re^2$,thus we have :

$$z(P) = \alpha x + \beta y + \gamma \tag{18}$$

Each Triple $V(x_i, y_i, z(P_i))$ satisfies equation 18, which leads to :

$$\begin{cases} z(P_1) = \alpha x_1 + \beta y_1 + \gamma \\ z(P_2) = \alpha x_2 + \beta y_2 + \gamma \\ z(P_3) = \alpha x_3 + \beta y_3 + \gamma \end{cases}$$

Therefore, if we consider the following points in : $\tilde{P}_1(x_1, y_1, z_1)$, $\tilde{P}_2(x_2, y_2, z_2)$, $\tilde{P}_3(x_3, y_3, z_3)$, $\tilde{P}(x, y, z) \in \Re^3$ it is clear that these points are coplanar (they lie in the same plane) D whose equation is given by (19a).

$$Ax + By + Cz + D = 0 \tag{19a}$$

$$Ax_1 + By_1 + Cz_1 + D = 0 \tag{19b}$$
$$Ax_2 + By_2 + Cz_2 + D = 0 \tag{19c}$$
$$Ax_3 + By_3 + Cz_3 + D = 0 \tag{19d}$$

$A,\ B,\ C,\ D \in \Re$, we can switch from equation (18) to (19a) by performing the substitutions : $\alpha = A,\ \beta = B,\ \gamma = D$ Now, after performing the following subtractions : (19a) − (19b) and (19c) − (19b) and (19d) − (19b) we derive the system of equations given below :

$$\begin{cases} A(x - x_1) + C(y - y_1) + C(z - z_1) = 0 \\ A(x_2 - x_1) + C(y_2 - y_1) + C(z_2 - z_1) = 0 \\ A(x_3 - x_1) + C(y_3 - y_1) + C(z_3 - z_1) = 0 \end{cases}$$

. A, B, C are the variables in the system of equations given above, it is obvious that the determinant : $\begin{vmatrix} x - x_1 & y - y_1 & z - z_1 \\ x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ x_3 - x_1 & y_3 - y_1 & z_3 - z_1 \end{vmatrix} = 0$

$$(x-x_1)\begin{vmatrix} y_2 - y_1 & z_2 - z_1 \\ y_3 - y_1 & z_3 - z_1 \end{vmatrix} - (y-y_1)\begin{vmatrix} x_2 - x_1 & z_2 - z_1 \\ x_3 - x_1 & z_3 - z_1 \end{vmatrix} + (z-z_1)\begin{vmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \end{vmatrix} = 0 \quad (20)$$

Once 20 has been resolved, it leads to formula (21) which can be used to interpolate z.

$$z = z_1 - (x - x_1)\frac{\begin{vmatrix} y_2 - y_1 & z_2 - z_1 \\ y_3 - y_1 & z_3 - z_1 \end{vmatrix}}{\begin{vmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \end{vmatrix}} + (y - y_1)\frac{\begin{vmatrix} x_2 - x_1 & z_2 - z_1 \\ x_3 - x_1 & z_3 - z_1 \end{vmatrix}}{\begin{vmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \end{vmatrix}} \quad (21)$$

From (21), we can easily compute the gradient of (18) :

$$\vec{\nabla} f \left(\frac{-\begin{vmatrix} y_2 - y_1 & z_2 - z_1 \\ y_3 - y_1 & z_3 - z_1 \end{vmatrix}}{\begin{vmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \end{vmatrix}}, \frac{\begin{vmatrix} x_2 - x_1 & z_2 - z_1 \\ x_3 - x_1 & z_3 - z_1 \end{vmatrix}}{\begin{vmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \end{vmatrix}} \right)$$

3.3 Application of the linear Interpolation: determination of $m(z),\ \nabla_z m(z),\ Q_z,\ \nabla_z Q_z$

Now let us go back to our previous problem: assuming we have a finite set of measures, how can we determine $m(z),\ Q_z$, at every position z of the plane? We recall that each measure corresponds to a position augmented with a mean vector and an inverse covariance matrix. $S = (z, m(z), Q_z)$.
After performing a Delaunay triangulation T_S on the S, we can use the simple linear interpolation method to find out the statistical parameters$m(z),\ Q_z$ everywhere.
We assume that each component $m(z)_i,\ i = 1..K$ of the vector $m(z)$ can be interpolated by a linear function that provides also an estimate of the gradient of $m(z)_i$ at the point z. In the same way each element $Q_{z_{ij}}$ of Q_z can be interpolated by a linear function that gives an estimate of the gradient of Q_z at z. It is necessary to underline the fact that the term *estimate* doesn't refer to any statistical estimation method.
Under these assumptions, it becomes easy to notice that interpolating a vector v means interpolating every component v_i of the vector and that interpolating a matrix a requires interpolating all its elements a_{ij}, therefore interpolating a

vector or matrix can be reduced to a finite number of scalar interpolation. In the same way estimating the differential of $m(z)$, $\nabla_z m(z)$ requires the estimation of all the gradients of the components$m(z)_i$ of $m(z)$; and that estimating of the of differential of $Q_z, \nabla_z Q_z$ requires the estimation of all the gradients of the elements of $Q_{z_{ij}}$. As we have already seen in the previous section, the computation of $m(z)$, Q_z is performed by two primitives: $Interpolate_{vector}$ Algorithm2 and $Interpolate_{matrix}$ Algorithm3.

.

Algorithm 2 $\boldsymbol{Interpolate_{vector}}$(v)

for $i = 1$ **to** $\dim(v)$ **do**
  $Interpolate_{scalar}(v_i)$
end for

Algorithm 3 $\boldsymbol{Interpolate_{matrix}}$(a)

for $i = 1$ **to** number of rows of a **do**
  for $j = 1$ **to** number of column of a **do**
    $Interpolate_{scalar}(a_{ij})$
  end for
end for

Algorithm 4 $\boldsymbol{Interpolate_{scalar}(P(x,y))}$

$\triangle(x_1,y_1,z_1),\ (x_2,y_1,z_2),\ (x_3,y_3),z_3 \longleftarrow \boldsymbol{Locate(P(x,y))}$

$$z \leftarrow z_1 - (x - x_1)\frac{\begin{vmatrix} y_2 - y_1 & z_2 - z_1 \\ y_3 - y_1 & z_3 - z_1 \end{vmatrix}}{\begin{vmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \end{vmatrix}} + (y - y_1)\frac{\begin{vmatrix} x_2 - x_1 & z_2 - z_1 \\ x_3 - x_1 & z_3 - z_1 \end{vmatrix}}{\begin{vmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \end{vmatrix}}$$

Return z

The computation of $\nabla_z m(z)$, $\nabla_z Q_z$ is performed by two primitives: $EstimateDifferential_{vector}$ Algorithm5 and $EstimateDifferential_{matrix}$ Algorithm6.

Algorithm 5 $\boldsymbol{EstimateDifferential_{vector}}$(v)

for $i = 1$ **to** $\dim(v)$ **do**
  $Estimate_{gradient}(v_i)$
end for

It is clear that calling all these primitives $Interpolate_{vector}$, $Interpolate_{matrix}$, $Interpolate_{scalar}$ requires performing Delaunay triangulation before, in the algorithm the primitive ***locate(P)*** is a query primitive which returns the triangle where $p(x, y)$ lies. To Compute the 2D-Delaunay triangulation we used CGAL [10], a Computational Geometry Algorithms Library that implements the concepts related to Delaunay triangulation and provides the necessary software components, objects and algorithms to perform two dimensional Delaunay computations : construction, queries (predicates evaluation).

Algorithm 6 $\boldsymbol{EstimateDifferential_{matrix}}$(a)

for $i = 1$ **to** number of rows of a **do**
  for $j = 1$ **to** number of column of a **do**
    $Estimate_{gradient}(a_{ij})$
  end for
end for

Algorithm 7 $\boldsymbol{Estimate_{gradient}(P(x,y))}$

$\triangle(x_1,y_1,z_1),\ (x_2,y_1,z_2),\ (x_3,y_3),z_3 \longleftarrow \boldsymbol{Locate(P(x,y))}$

$$grad \leftarrow \vec{\nabla} f \left(\frac{-\begin{vmatrix} y_2-y_1 & z_2-z_1 \\ y_3-y_1 & z_3-z_1 \end{vmatrix}}{\begin{vmatrix} x_2-x_1 & y_2-y_1 \\ x_3-x_1 & y_3-y_1 \end{vmatrix}}, \frac{\begin{vmatrix} x_2-x_1 & z_2-z_1 \\ x_3-x_1 & z_3-z_1 \end{vmatrix}}{\begin{vmatrix} x_2-x_1 & y_2-y_1 \\ x_3-x_1 & y_3-y_1 \end{vmatrix}} \right)$$

Return *grad*

4 Model Refinement

We assume that we have K anchored points, each signal is sent from a specific AP_i, we also assume that there is no direct interaction between the different signals, which statistically leads to a null covariance between the components of the signals level vector and which implies a diagonal covariance matrix.

Therefore Q_z^{-1} is given by:

$$Q_z^{-1} = \begin{bmatrix} \delta_1(z) & . & . & \cdots & . \\ . & \delta(z)_2 & . & \cdots & . \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \vdots & \cdots & \delta(z)_K \end{bmatrix}$$

Since all the signals are random, we must have $\forall i, \delta(z)_i > 0$

This yields the following inverse covariance matrix Q_z :

$$Q_z = \begin{bmatrix} \lambda(z)_1 & . & . & \cdots & . \\ . & \lambda(z)_2 & . & \cdots & . \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \vdots & \cdots & \lambda(z)_K \end{bmatrix}$$

Where $\lambda(z)_i = \frac{1}{\delta_i}(z)$ and $\forall i, \lambda(z)_i > 0$. Therefore, $\nabla\lambda(z)_i = -\frac{\nabla\delta(z)_i}{\delta(z)_i^2}$
Consequently :

$$\nabla Q_z = \begin{bmatrix} -\frac{\nabla\delta(z)_1}{\delta(z)_1^2} & . & . & \cdots & . \\ . & -\frac{\nabla\delta(z)_2}{\delta(z)_2^2} & . & \cdots & . \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \vdots & \cdots & -\frac{\nabla\delta(z)_K}{\delta(z)_K^2} \end{bmatrix}$$

Let us now consider the computations of: $\nabla_z l(z,v)$ and R_z

$$\nabla_z l(z,v) = \sum_{i=1}^{K}\left(\frac{1}{2}\Big((v-m(z)_i)^2-\delta(z)_i\Big)\nabla_z\lambda(z)_i - \lambda(z)_i(v-m(z)_i)\nabla_z m(z)_i\right)$$

$$\nabla_z l(z,v) = \sum_{i=1}^{K}\left(\frac{1}{2}\Big(\frac{1}{\delta(z)_i}-(v-m(z)_i)^2\Big)\nabla_z\delta(z)_i - \Big(\frac{1}{\delta(z)_i}(v-m(z)_i)\Big)\nabla_z m(z)_i\right) \tag{22}$$

$$R_z = -\sum_{i=1}^{K}\Big(\frac{1}{2}\frac{1}{\delta(z)_i^2}\delta(z)_i\delta(z)_i^T + \frac{1}{\delta(z)_i}\nabla_z m(z)_i m(z)_i^T\Big)$$

Since $\nabla m(z)_i = \begin{pmatrix} \frac{\partial m(z)_i}{\partial x} \\ \frac{\partial m(z)_i}{\partial y} \end{pmatrix}$

$$R_z = -\sum_{i=1}^{K}\begin{bmatrix} \frac{1}{2}\frac{1}{\delta(z)_i^2}(\frac{\partial\delta(z)_i}{\partial x})^2 + \frac{1}{\delta(z)_i}(\frac{\partial m(z)_i}{\partial x})^2 & \frac{1}{2}\frac{1}{\delta(z)_i^2}\frac{\partial\delta(z)_i}{\partial x}\frac{\partial\delta(z)_i}{\partial y} + \frac{1}{\delta(z)_i}\frac{\partial m(z)_i}{\partial x}\frac{\partial m(z)_i}{\partial y} \\ \frac{1}{2}\frac{1}{\delta(z)_i^2}\frac{\partial\delta(z)_i}{\partial x}\frac{\partial\delta(z)_i}{\partial y} + \frac{1}{\delta(z)_i}\frac{\partial m(z)_i}{\partial x}\frac{\partial m(z)_i}{\partial y} & \frac{1}{2}\frac{1}{\delta(z)_i^2}(\frac{\partial\delta(z)_i}{\partial y})^2 + \frac{1}{\delta(z)_i}(\frac{\partial m(z)_i}{\partial y})^2 \end{bmatrix} \tag{23}$$

4.1 Some useful properties of R and its inverse R^{-1}

From (23) we derivate the following equality:

$$R_z = -\begin{bmatrix} \sum_{i=1}^{K}\left(\frac{1}{2}\frac{1}{\delta(z)_i^2}(\frac{\partial\delta(z)_i}{\partial x})^2 + \frac{1}{\delta(z)_i}(\frac{\partial m(z)_i}{\partial x})^2\right) & \sum_{i=1}^{K}\left(\frac{1}{2}\frac{1}{\delta(z)_i^2}\frac{\partial\delta(z)_i}{\partial x}\frac{\partial\delta(z)_i}{\partial y} + \frac{1}{\delta(z)_i}\frac{\partial m(z)_i}{\partial x}\frac{\partial m(z)_i}{\partial y}\right) \\ \sum_{i=1}^{K}\left(\frac{1}{2}\frac{1}{\delta(z)_i^2}\frac{\partial\delta(z)_i}{\partial x}\frac{\partial\delta(z)_i}{\partial y} + \frac{1}{\delta(z)_i}\frac{\partial m(z)_i}{\partial x}\frac{\partial m(z)_i}{\partial y}\right) & \sum_{i=1}^{K}\left(\frac{1}{2}\frac{1}{\delta(z)_i^2}(\frac{\partial\delta(z)_i}{\partial y})^2 + \frac{1}{\delta(z)_i}(\frac{\partial m(z)_i}{\partial y})^2\right) \end{bmatrix} \tag{24}$$

After performing the following substitutions :

$$\begin{aligned} S_1 &= \sum_{i=1}^{K}\left(\frac{1}{2}\frac{1}{\delta(z)_i^2}(\frac{\partial\delta(z)_i}{\partial x})^2 + \frac{1}{\delta(z)_i}(\frac{\partial m(z)_i}{\partial x})^2\right) \\ S_2 &= \sum_{i=1}^{K}\left(\frac{1}{2}\frac{1}{\delta(z)_i^2}(\frac{\partial\delta(z)_i}{\partial y})^2 + \frac{1}{\delta(z)_i}(\frac{\partial m(z)_i}{\partial y})^2\right) \\ S &= \sum_{i=1}^{K}\left(\frac{1}{2}\frac{1}{\delta(z)_i^2}\frac{\partial\delta(z)_i}{\partial x}\frac{\partial\delta(z)_i}{\partial y} + \frac{1}{\delta(z)_i}\frac{\partial m(z)_i}{\partial x}\frac{\partial m(z)_i}{\partial y}\right) \end{aligned} \tag{25}$$

Thus :

$$R_z = -\begin{bmatrix} S_1 & S \\ S & S_2 \end{bmatrix}$$

we can easily notice that R is *symmetric*, let us compute its inverse R_z^{-1} :

$$R_z^{-1} = \frac{1}{\det(R)}Com(R)^T$$

where $Com(R)^T$ is the transpose of the adjugate of R_z, after performing a simple calculus we have :

$$R_z^{-1} = \frac{1}{S^2 - S^2S_1}\begin{bmatrix} S_2 & S \\ S & S_1 \end{bmatrix} \tag{26}$$

(25) shows that R_z^{-1} is also *symmetric*

4.1.1 Singularity of the matrix R

Since the computation of the matrix R_z, represents an important step in our algorithm. We should investigate the singularity aspect of the Matrix R, because we need to know whether R_z remains invertible for every iteration.But before, we have to take advantage of the linear expression of each $m(z)_i$ due the linear interpolation.Since $m(z)_i,\ \delta(z)_i$ are interpolated at every position z with a linear functions, we can approximate them as follows:

$$m(z)_i = \alpha_i^T z + \beta_i \tag{27}$$

$$\delta(z)_i = \Gamma_i^T z + \eta_i \tag{28}$$

Where $m(z)_i$ is the mean of the signal strength which is received from the anchored point AP_i. $\delta(z)_i$ is the variance of the signal strength which is received AP_i we recall that :

$$\nabla_z m(z) = \alpha_i = \begin{pmatrix} \alpha_{i_x} \\ \alpha_{i_y} \end{pmatrix} \tag{29}$$

$$\nabla_z \delta(z) = \Gamma_i = \begin{pmatrix} \Gamma_{i_x} \\ \Gamma_{i_y} \end{pmatrix} \tag{30}$$

(29) and (30) represent respectively the gradients of $m(z)_i,\ \delta(z)_i$ with a respect to z while $\beta_i,\ \eta_i \in \Re$ are scalars.After substituting (27) in (25) we obtain :

$$\begin{aligned} S_x^{(K)} &= \sum_{i=1}^{K} \Big(\lambda(z)_i (\alpha_{i_x})^2 + \mu(z)_i (\Gamma_{x_i})^2 \Big) \\ S_{xy}^{(K)} &= \sum_{i=1}^{K} \Big(\lambda(z)_i \alpha_{i_x} \alpha_{i_y} + \mu(z)_i \Gamma_{x_i} \Gamma_{y_i} \Big) \\ S_y^{(K)} &= \sum_{i=1}^{K} \Big(\lambda(z)_i (\alpha_{i_y})^2 + \mu(z)_i (\Gamma_{y_i})^2 \Big) \end{aligned} \tag{31}$$

Where $\lambda(z)_i = \frac{1}{\delta(z)_i}$ and $\mu(z)_i = \frac{1}{2}\frac{1}{\delta(z)_i^2}$. The number of anchored points must be at least equal to two $K \geq \dim(P)$, which is the dimension of the plane.Generally the number of anchored points should be greater or equal than two ≥ 2 . Since $S_1,\ S,\ S_2$ are sums that depend on the number of anchored points K, we can ask the following questions :

Is there any relation between the singularity of R_z and the number of anchored points K? and if so under which conditions R does remain invertible?
The following result is very useful to find out the required conditions for the inversion of R

Proposition 1 $\forall K \geq 2,\ \det(R_{z_K}) = \sum_{i=2}^{K} \sum_{j=1 j<i} \big(\lambda(z)_i \lambda(z)_j \det(\alpha_i, \alpha_j)^2 + \mu(z)_i \mu(z)_j \det(\Gamma_i, \Gamma_j)^2 \big) + \sum_{l=1}^{K} \sum_{m=1}^{K} \lambda(z)_l \mu(z)_m \det(\alpha_l, \Gamma_m)^2$

$\mathfrak{Proof}$: We proof this result by induction. First we demonstrate that the statement holds for the case $n = K = 2$ after that we demonstrate that if the

statement holds for the case n, then it also holds for the case $n+1$. Let us verify that the statement holds for $n=2$

$$R_{z_K} = -\begin{bmatrix} \sum_{j=1}^{2}\left(\mu(z)_j\Gamma_{j_x}^2 + \lambda(z)_j\alpha_{j_x}^2\right) & \sum_{j=1}^{2}\left(\mu(z)_1\Gamma_{j_x}\Gamma_{j_y} + \lambda(z)_j\alpha_{j_x}\alpha_{j_y}\right) \\ \sum_{j=1}^{2}\left(\mu(z)_j\Gamma_{j_x}\Gamma_{j_y} + \lambda(z)_j\alpha_{j_x}\alpha_{j_y}\right) & \sum_{j=1}^{2}\left(\mu(z)_j\Gamma_{j_y}^2 + \lambda(z)_j\alpha_{j_y}^2\right) \end{bmatrix}$$

$$\begin{aligned} \det(R_{z_K}) = &\left(\sum_{j=1}^{2}(\mu(z)_j\Gamma_{j_x}^2 + \lambda(z)_j\alpha_{j_x}^2)\sum_{j=1}^{2}(\mu(z)_j\Gamma_{j_y}^2 + \lambda(z)_j\alpha_{j_y}^2)\right) \\ &- \left(\sum_{j=1}^{2}(\mu(z)_j\Gamma_{j_x}\Gamma_{j_y} + \lambda(z)_j\alpha_{j_x}\alpha_{j_y}))\right)^2 \end{aligned} \tag{32}$$

$$\det(R_{z_K}) = S_xS_y - S_{xy}^2 \tag{33}$$

we denote:

$$\begin{aligned} S_x^{(2)} &= \mu(z)_1\Gamma_{1_x}^2 + \lambda(z)_1\alpha_{1_x}^2 + \mu(z)_2\Gamma_{2_x}^2 + \lambda(z)_2\alpha_{2_x}^2 \\ S_y^{(2)} &= \mu(z)_1\Gamma_{1_y}^2 + \lambda(z)_1\alpha_{1_y}^2 + \mu(z)_2\Gamma_{2_y}^2 + \lambda(z)_2\alpha_{2_y}^2 \\ S_{xy}^{(2)} &= \left(\mu(z)_1\Gamma_{1_x}\Gamma_{1_y} + \lambda(z)_1\alpha_{1_x}\alpha_{1_y} + \mu(z)_2\Gamma_{2_x}\Gamma_{2_y} + \lambda(z)_2\alpha_{2_x}\alpha_{2_y}\right)^2 \end{aligned}$$

.

After developing (33) and simplifying some terms, we finally obtain :

$$\begin{aligned} \det(R_{z_K}) = &\lambda(z)_1\lambda(z)_2\det(\alpha_1,\alpha_2)^2 + \lambda(z)_1\mu(z)_1\det(\alpha_1,\Gamma_1)^2 \\ &+\lambda(z)_1\mu(z)_2\det(\alpha_1,\Gamma_2)^2 + \lambda(z)_2\mu(z)_1\det(\alpha_2,\Gamma_1)^2 \\ &+\lambda(z)_2\mu(z)_2\det(\alpha_2,\Gamma_2)^2 + \mu(z)_1\mu(z)_2\det(\Gamma_1,\Gamma_2)^2 \end{aligned}$$

Now we demonstrate that if the statement holds for the case n, then it also holds for the case $n+1$, if the statement holds for n we then we have :

$$\begin{aligned} \det(R_{z_n}) = &\sum_{i=2}^{n}\sum_{j=1 j<i}\left(\lambda(z)_i\lambda(z)_j\det(\alpha_i,\alpha_j)^2 + \mu(z)_i\mu(z)_j\det(\Gamma_i,\Gamma_j)^2\right) \\ &+\sum_{l=1}^{n}\sum_{m=1}^{n}\lambda(z)_l\mu(z)_m\det(\alpha_l,\Gamma_m)^2 \end{aligned}$$

let us demonstrate that the statement holds for $n+1$;

$$R_{z_{n+1}} = \begin{bmatrix} S_{n_x} + t_{(n+1)_x} & S_{n_{xy}} + t_{(n+1)_{xy}} \\ S_{n_{xy}} + t_{(n+1)_{xy}} & S_{n_y} + t_{(n+1)_y} \end{bmatrix}$$

where :

$$\begin{aligned} S_x^{(n)} &= \sum_{j=1}^{n} (\mu(z)_j \Gamma_{j_x}^2 + \lambda(z)_j \alpha_{j_x}^2) \\ S_y^{(n)} &= \sum_{j=1}^{n} (\mu(z)_j \Gamma_{j_y}^2 + \lambda(z)_j \alpha_{j_y}^2) \\ S_{xy}^{(n)} &= \sum_{j=1}^{n} (\mu(z)_j \Gamma_{j_x} \Gamma_{j_y} + \lambda(z)_j \alpha_{j_x} \alpha_{j_y}) \end{aligned} \tag{34}$$

and where

$$\begin{aligned} t_{(n+1)_x} &= \mu(z)_{n+1} \Gamma_{n+1_x}^2 + \lambda(z)_{n+1} \alpha_{n+1_x}^2 \\ t_{(n+1)_y} &= \mu(z)_{n+1} \Gamma_{n+1_y}^2 + \lambda(z)_{n+1} \alpha_{n+1_y}^2 \\ t_{(n+1)_{xy}} &= \mu(z)_{n+1} \Gamma_{n+1_x} \Gamma_{n+1_y} + \lambda(z)_{n+1} \alpha_{n+1_x} \alpha_{n+1_y} \end{aligned} \tag{35}$$

$$R_{z_{n+1}} = (S_{n_x} + t_{(n+1)_x}) * (S_{n_y} + t_{(n+1)_y}) - (S_{n_{xy}} + t_{(n+1)_{xy}})^2$$

which yields :

$$\begin{aligned} & R_{z_{n+1}} = S_{n_x} S_{n_y} + S_{n_x} t_{(n+1)_y} + t_{(n+1)_x} S_{n_y} + t_{n+1_x} t_{(n+1)_y} \\ & - (S_{n_{xy}}^2 + 2 S_{n_{xy}} t_{(n+1)_{xy}} + t_{(n+1)_{xy}}^2) \\ & = S_{n_x} S_{n_y} - S_{n_{xy}}^2 + S_{n_x} t_{(n+1)_y} + t_{(n+1)_x} S_{n_y} - 2 S_{n_{xy}} t_{(n+1)_{xy}} - t_{(n+1)_{xy}}^2 \\ & = R_{z_n} + S_{n_x} t_{(n+1)_y} + t_{(n+1)_x} S_{n_y} - 2 S_{n_{xy}} t_{(n+1)_{xy}} - t_{(n+1)_{xy}}^2 \end{aligned} \tag{36}$$

After developing and simplifying terms of (36) we get :

$$det(R_{z_{n+1}}) = R_{z_n} + \sum_{i=1}^{n} \Big(\lambda(z)_{n+1} \lambda(z)_i \det(\alpha_{n+1}, \alpha_i)^2 + \mu(z)_{n+1} \mu(z)_i \det(\Gamma_{n+1}, \Gamma_i)^2 \tag{37}$$

$$+ \lambda(z)_{n+1} \mu(z)_i \det(\alpha_{n+1}, \Gamma_i)^2 + \mu(z)_{n+1} \lambda(z)_i \det(\Gamma_{n+1}, \alpha_i)^2 \Big) \tag{38}$$

$$+ \lambda(z)_{n+1} \mu(z)_{n+1} \det(\alpha_{n+1}, \Gamma_{n+1})^2 \tag{39}$$

which finally Leads to:

$$\begin{aligned} \det(R_{z_{n+1}}) &= \sum_{i=2}^{n+1} \sum_{j=1 j<i} \left(\lambda(z)_i \lambda(z)_j \det(\alpha_i, \alpha_j)^2 + \mu(z)_i \mu(z)_j \det(\Gamma_i, \Gamma_j)^2 \right) \\ &+ \sum_{l=1}^{n+1} \sum_{m=1}^{n+1} \lambda(z)_l \mu(z)_m \det(\alpha_l, \Gamma_m)^2 \end{aligned}$$

REMARK 1 *We can easily notice that R_{z_k} is always positive :*

$$R_{z_k} \geq 0, \ \forall K \geq 2 \tag{40}$$

5 Simulations and Results

5.1 Experimental Setup

5.1.1 introduction

First we will describe the setup of our experimentations, then we will detail the flow processing of our application prototype.After that we will introduce our simulation scenarios by describing the simulated motions which we used.

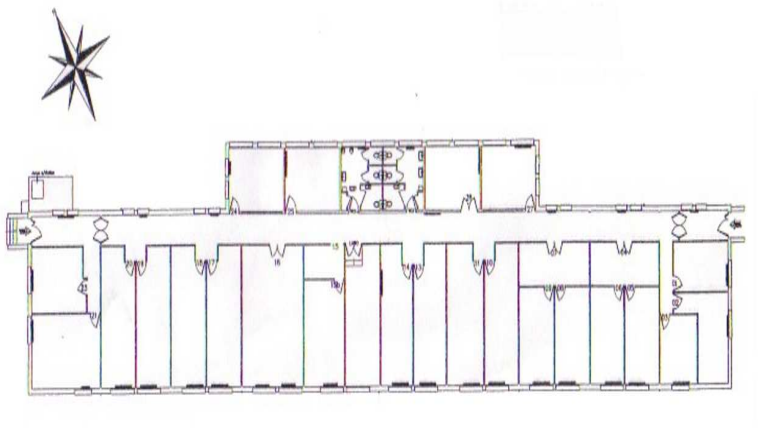


Figure 5: Buidling 17

5.2 The Building

We carried out our experiments in building 17 of the INRIA Rocquencourt center, which is the building of the (HIPERCOM Project).

One of the main features of this kind of buildings is their main corridor (Figure 5), It is a long corridor, which has more offices on one side than the other.
We are interested in simulating different motions in the first quarter of the corridor.To achieve this purpose the locations of our Anchored Points (Figure 6) are chosen to provide a persistent coverage of the first quarter of the corridor.
Our *localization application* requires building the signal map of building 17, which has six OLSR[11] routers, that are used as anchored points.
Each router is configured in Ad-Hoc mode and programmed to send periodical UDP signals. We would like to underscore that we could configure these routers in the infrastructure mode, but in this case we would rather use beacons, we recall that beacons are a kind of management frames which are used for network identification, broadcasting network capabilities and synchronization[].
As is stated above, the locations of the anchored points (Figure 6) were chosen to provide a coverage throughout the corridor of the building especially the first quarter.
We performed our experiments on a laptop PC Dell Latitude D610 which is equipped with a Planex WLAN card with the Atheros 5212 chip-set; our host PC is running a 2.6.27-14 generic Linux kernel.

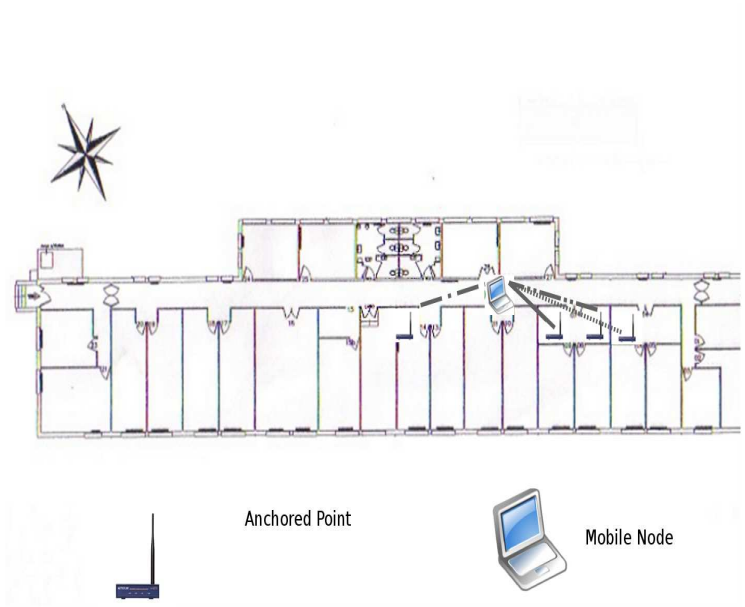


Figure 6: Location of the Anchored Points

5.3 Operating System and software development environment of the localization application

Each router is running a 2.4.19-Router Linux kernel.To construct the signal map, which is shown in(Figure 7), we have developed our own Wifi packet sniffer.This sniffer is implemented in C++ and developed with g++ 4.3.2.It also uses the primitives of the PCAP library; we recall that PCAP is a capture library that provides a high level interface to packet capture systems[12].
MadWifi's[13] driver is used to communicate with the WLAN card, this driver provides information about the parameters of reception,including:

- The power in decibels of the signal strength received by the antenna.
- The power in decibels of noise.

Our Localization application is written in C++ and developed with g++ 4.3.2, we used the C++ Library Newmat [14] to perform Matrix operations (inversing Matrices, computing Kronecker product) and CGAL[10] to perform geometrical algorithm operations (Delaunay triangulation, $2D$ point predicates queries).

5.4 Signal map:

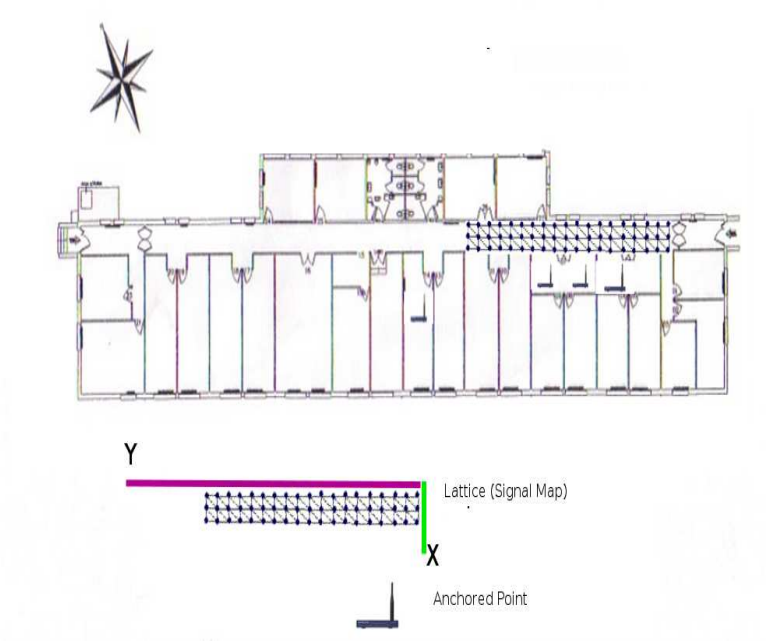


Figure 7: Signal Map

The signal map stores compact information of statistical parameters of the signals received.Moreover the signal map is inherent to our approach since we consider the received signals to be random and since the storage of statistical information for all the positions is highly expensive.
As is mentioned in subsection 3.3, the signal map is a finite collection of positions z which are augmented with statistical information $m(z)$, Q_z that depend on z.We can consider the signal map as a finite set: Signal Map=$\{(position_j, parameters_j)\}$ where : $parameters_j = (m(position_j),\ Q_{position_j})$.
In our case Figure(7)$position_j$ are regularly spaced generating an organized lattice. The distance that separates each element from its vertical neighbor (along the X axis) is constant = 0.6 meter while the horizontal distance that separates each element from its horizontal neighbor (along the Y axis) is constant = 0.9 meter.
Our lattice contains 57 points organized in three rows of nineteen points. To estimate the different $m(z)_i$ composing $m(z)$, we need to collect samples E_i that contain observations which correspond to the values of the signal strength received from th anchored point Ap_i.
The size N of these samples is constant, in our case we chose $N = 150$; the same samples E_i are used to estimate the $\lambda(z)_i$ via computing : $\lambda(z)_i = \frac{N-1}{N\delta(z)_i}$ where $\delta(z)_i$ is the variance of E_i.
The Delaunay triangulation of the elements of the *Signal map* is shown in (figure); the signal map is coupled with the simple linear interpolation method to interpolate linearly $m(z)$,Q_z ,$\nabla_z m(z)$ at any position.

5.5 The application prototype (flow processing)

The flow processing of the Localization application is shown in (Figure 8).Each block of the processing flow corresponds to a processing phase of our *Localization application.* Since our application can be run in either simulated or real mode, two alternative phases are necessary to the flow processing:
Simulated mode: In this mode the path we want to track $C(t) = (x(t), y(t))$ is generated using its mathematical equation of motion, in this case the received signal level vector is a Gaussian vector which is simulated using the parameters $m(z')$, $Q_{z'}$, where the coordinates $z'(x', y')$ are generated using the mathematical equation of motion.
Real Mode: The signal level vector is captured with a sniffing primitive that uses PCAP and information provided by MADWIFI. The core of this primitive is similar to the core of the sniffer used to establish the signal map[]. Statistical parameters Interpolation: during this phase we compute the different parameters ($m(z)$, $\nabla_z m(z)$, Q_z) which are required for the filtering phase; the computation of these parameters is performed by using the measures that constitute our signal map.
Filtering: This phase constitutes the main block of the the flow process whereby our filtering algorithm is performed.
Display:, In the display phase we plot the predicted path and the real path.
Signal map: As mentioned above, the signal map is combined with the interpolation primitives seen in []to constitute a data base.This DB provides both *the Filtering phase* and *the Simulation of capture of signals phase* with the required statistical parameters at any position z inside the first quarter corridor.

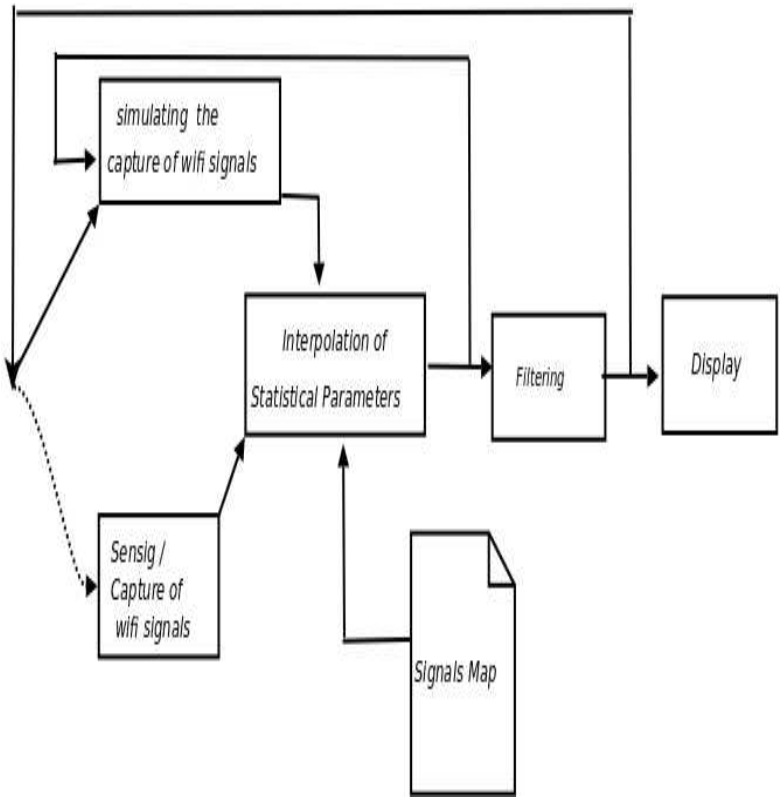


Figure 8: Flow processing of the localization application

5.6 Simulations and results :

To validate our algorithm, we first test it on uniform linear motions such as: simple rectilinear motions and complex rectilinear motion, then we extend our test to simple uniform circular motion.We would like to highlight that by a complex rectilinear motion we mean a motion which is composed of a successive phases where each phase corresponds to a simple uniform motion. All these uniform motions implie a constant velocity or constant angular velocity in addition, both of which are generated from their parametric representation $z(x(t), y(t))$. The entire time of simulation is divided into finite constant time steps where each step corresponds to one second.

5.6.1 Simulating the capture of wifi signals

Now let us detail the phase which consists of simulating the capture of WIFI signals. The main goal of this phase is to simulate the sensing process which means the capturing the vector of the received signals strengths we first need to characterize the path we want to simulate by a parametric equations.We call this path the real path while we call the path which is estimated by our filter the estimated path. We denote the real path by $C = Z(X(t), Y(t))$ and we generate the positions $Z(t) = (X(t), Y(t))$ using the parametric equations of C:

$$\begin{cases} X(t) = f(t) \\ Y(t) = g(t) \end{cases}$$

At each time step we compute $Z(t) = (X(t), Y(t))$, by which we can interpolate $m(Z(t))$, $Q_{Z(t)}$.Finally we simulate the vector of the received signals strength by simulating the Gaussian vector of the mean $m(Z(t)$ and the inverse matrix covariance $Q_{Z(t)}$.

5.6.2 Quantifying the error:

To quantify the error between the real path and the estimated path which is predicted by our filter, we have introduced the following measure of error :

$$E_t = \int_0^T \|\hat{Z(t)} - Z(t)\| dt$$

where $Z(t)$ is the real position and $\hat{Z(t)}$ is the estimated position.The global cumulative error is equal to the sum of the difference between the real and the estimated position during the simulation time T. Where $\|..\|$ is the euclidean norm. Each simulation result, is provided with two figures ; the first figure shows the real trajectory vs the trajectory estimated by our algorithm; the second figure shows the cumulative error -iteration evolution.

5.7 Results:

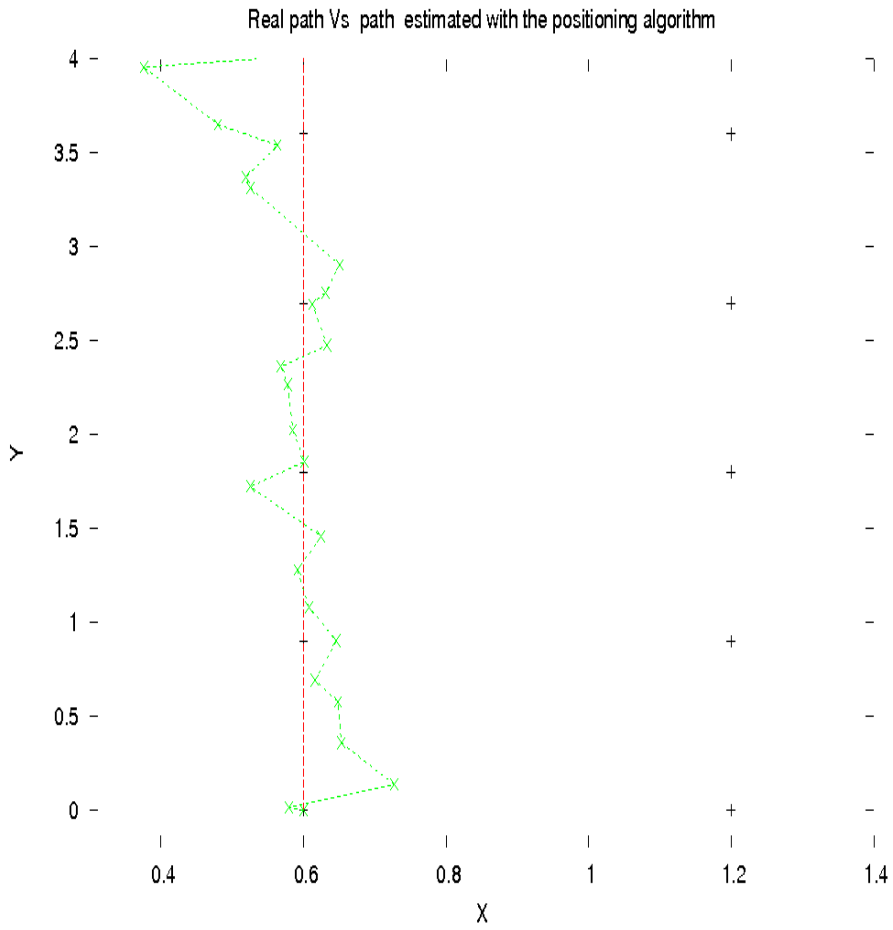
A time step corresponds to performing one iteration of our algorithm ; in our simulations we chose this time step $\zeta(t) = 1$ second.

5.7.1 Vertical trajectory:

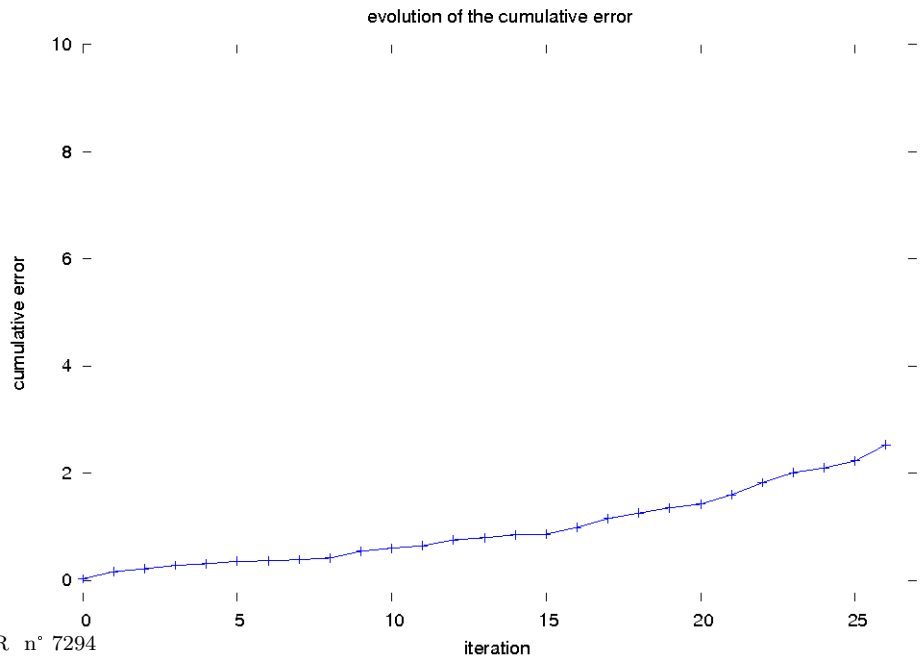
Assume we have a target or a mobile node that moves in a vertical trajectory with a uniform motion. The vertical trajectory is generated using the following equation:

$$\begin{cases} Z(0) = (X(0), Y(0)) \\ X(t) = X(0) = constant \\ Y(t) = Y(t-1) + W_y + \forall t \geq 1 \end{cases}$$

Now let us distinguish two cases: The first case: the trajectory of the motion vm_1 (Figure 9(a)) corresponds to the vertical line segment that starts from point $z_0 = (0.6, 0.0)$ and joins point $z_f(0.6, 4.1)$, this line segment is obtained with $(W_y = 0.18)$ and $Z(0) = (0.6, 0.0)$, with a these parameters we have a linear (a linear velocity) of $d_{vm_1} = 0.18$;we notice that, the cumulative error remains almost constant during the first fifteen iterations, the error starts to increase significantly figure 9(a) . After running 27 iterations, we get a cumulative error of $E_{vm_1} = 3.82meter$.



(a) Real trajectory vs predicted trajectory



(b) Cumulative error-iteration evolution

Figure 9: Result 1

5.8 Complex uniform rectilinear Motion:

As already stated above, this motion (Figure **??**) is composed of two or more phases, in this case we consider two phases which implies two trajectories: The trajectory of the first motion is generated by the equation:

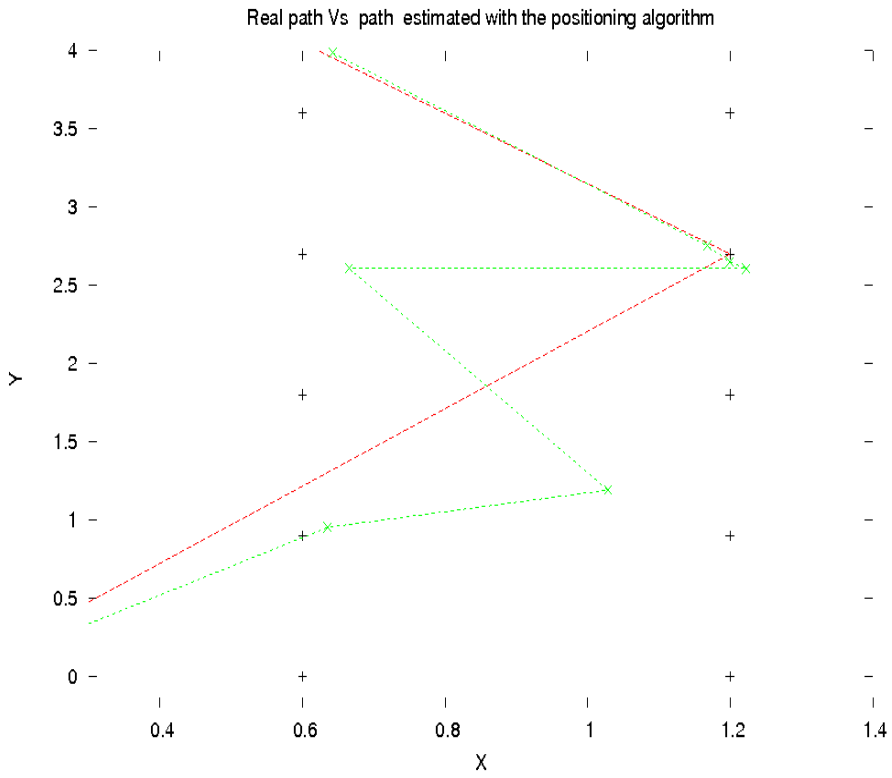
$$\begin{cases} Z_1(0) = (X_{1_0}, Y_{1_0}) \\ X_1(t) = X_1(t-1) + W_{1_x} \\ Y_1(t) = Y_1(t-1) + W_{1_y} \end{cases}$$

With $W_{1_y} = \alpha_1 W_{1_x}$ +The trajectory of the second motion is generated by the equation:

$$\begin{cases} Z_2(0) = (X_{2_0}, Y_{2_0}) \\ X_2(t) = X_2(t-1) + W_{2_x} \\ Y_2(t) = Y_2(t-1) + W_{2_y} \end{cases}$$

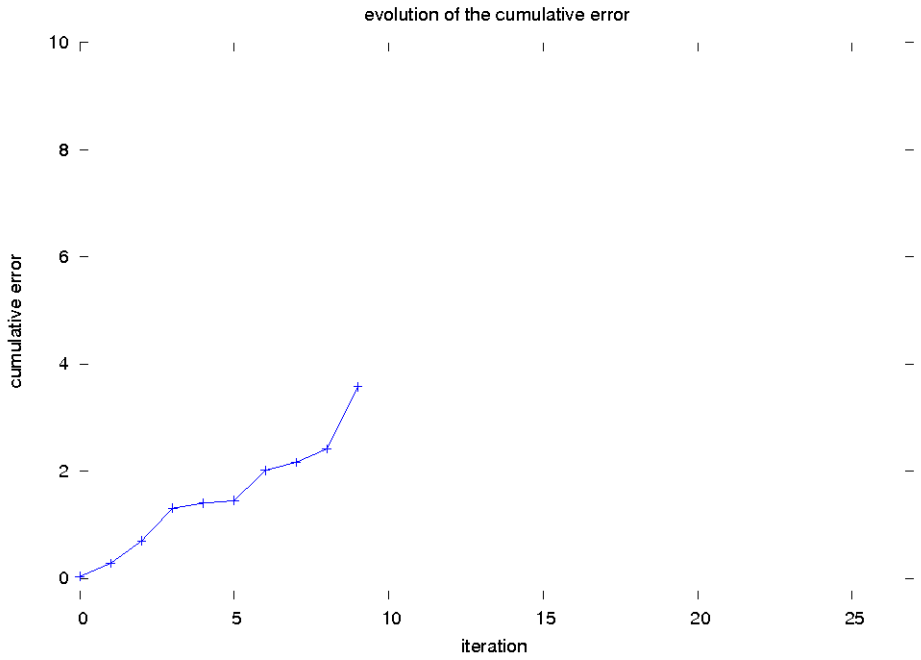
where :$W_{2_y} = \alpha_2 W_{2_x}$ and $W_{2_x} = -W_{1_x},\ \alpha_2 = -\alpha_1$.

The first motion describes an oblique line segment that starts from:$Z_{1_0} = (0.12, 0.12)$ and reaches $Z_{1_f} = (1.2, 2.7)$, $(\alpha_1 = 2.47,\ W_{1_x} = 0.25)$ with a shifting of $d_{cm1} = 0.47 meter/second$.
The second motion also describes an oblique line segment, that starts from: $Z_{2_0} = Z_{1_f}$ and reaches $Z_{2_f} = (0.6, 4.0)$, $(\alpha_2 = -2.47,\ W_{2_x} = -0.25 meter)$ with a shifting of $d_{cm2} = 0.47\ meter/second$. The Figure 10(b) shows the evolution of cumulative error curve, unlike the first result, the cumulative error curve, increases immediately to reach the value of $4 meter$ after ten iterations , this might be due to the complexity of the motion.



ig:Cm-a

(a) Real trajectory vs predicted trajectory



(b) Cumulative error-iteration evolution

Figure 10: Result 2

6 Conclusion

We have designed and developed a filtering algorithm which does not exploit any information about the motion model.
We have shown that the computations generated by our filtering algorithm cannot degenerate as long as the target remains inside the signal map; this is mainly due to the nonsingular character of the key matrix R which cannot be singular.

References

[1] H. D. P. B. J. L. Hui Liu, "Survey of wireless indoor positioning techniques and systems," *IEEE*, pp. 1067–1080, 2007.

[2] T. R. P. M. H. T. P. Misikangas and J. Sievanen, "A probabilistic approach to wlan user location estimation," *International Journal of wireless Networks*, vol. 9, pp. 155–162, 2002.

[3] Y. C. H. Kobayashi, "Signal strength based indoor geolocation," *IEEE*, pp. 436–439, 2002.

[4] *IEEE INFOCOM 2000, Vol. 2, Tel-Aviv, Israel (March 2000): 775-784*, 2000.

[5] W. A. A. N. S. Y. J. Wan, "Your 802.11 wireless network has no clothes," tech. rep., Department of Computer Science University of Maryland, 2001.

[6] H. H. S. Kokichi, "Vornoi-based interpolation with higher continuity," *Computational Geometry Theory and application*, pp. 242–243, 2000.

[7] H. H. S. Kokichi, "Improving continuity of voronoi -based interpolation over delaunay spheres," *Computational Geometry Theory and application*, pp. 168–169, 2001.

[8] F. G. H. J. K. Myung-Soo, *Handbook of Comptuer Aided Geometric Design*. 2002.

[9] B. T. F. G. F. H. D. U. George, "Natural neighbor extrapolation using ghost points," *Computer-Aided Design*, 2008.

[10] *CGAL, http::://www.CGAL.org*.

[11] *OLSR, http://www.olsr.org/*.

[12] *PCAP, http://www.tcpdump.org/*.

[13] *MADWIFI, http://madwifi-project.org/*.

[14] *NewMat, http://ideas.repec.org/c/cod/ccplus/newmat.html*.

Contents

Centre de recherche INRIA Paris – Rocquencourt
Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399